CS202 Assignment #1

Purpose:
Refresh concepts from CS 135 including functions, file input, vectors, and introduce string streams.

Introduction:
YouTube video Introducing the assignment https://youtu.be/WlMMRv4ilAo

Supplied:
main.cpp program is provided as the driver for the functions that you will write. Students should write the functions and test the program. The *main()* function should not be modified. The database file, AISData.csv, is supplied as a compressed zip file on Canvas and is available on Sally uncompressed. If the compressed file is downloaded from Canvas, it must be uncompressed before using.

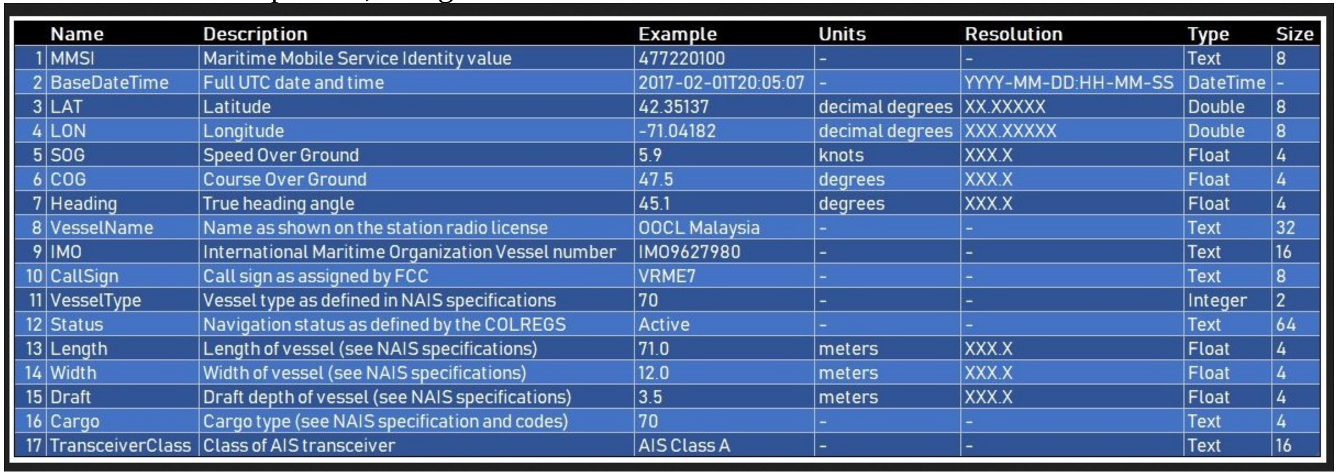
Expected Submissions:
main.cpp with the students functions incorporated. The file should be submitted online via the class Canvas website. **No late submissions will be accepted.**

Problem:
This program will read in a large US Coast Guard database of ship position information from the Automatic Identification System (AIS)[1]. The student provided functions will:

1. Prompt the user for the filename and open the file.
2. Read in data from the file into the vector database.
3. Prompt the user for a name/phrase to find in the vessel names.
4. Build a vector that contains unique vessels that contain the name/phrase.
5. If the number of vessels found is not zero, prompt to see if the user wants to display the vessel records whose names contain the phrase. If yes show records.
    1. Prompt the user to see if they want to calculate the distance traveled? If yes, do calculation and display distance.
6. Go to step 3 until quit is requested.

[1] https://navcen.uscg.gov/?pageName=AISReferences
[2] Data downloaded from https://marinecadastre.gov/ais/

File Data Format
The file is comma separated, strings in the order listed below.



	Name	Description	Example	Units	Resolution	Type	Size
1	MMSI	Maritime Mobile Service Identity value	477220100	-	-	Text	8
2	BaseDateTime	Full UTC date and time	2017-02-01T20:05:07	-	YYYY-MM-DD:HH-MM-SS	DateTime	-
3	LAT	Latitude	42.35137	decimal degrees	XX.XXXXX	Double	8
4	LON	Longitude	-71.04182	decimal degrees	XXX.XXXXX	Double	8
5	SOG	Speed Over Ground	5.9	knots	XXX.X	Float	4
6	COG	Course Over Ground	47.5	degrees	XXX.X	Float	4
7	Heading	True heading angle	45.1	degrees	XXX.X	Float	4
8	VesselName	Name as shown on the station radio license	OOCL Malaysia	-	-	Text	32
9	IMO	International Maritime Organization Vessel number	IMO9627980	-	-	Text	16
10	CallSign	Call sign as assigned by FCC	VRME7	-	-	Text	8
11	VesselType	Vessel type as defined in NAIS specifications	70	-	-	Integer	2
12	Status	Navigation status as defined by the COLREGS	Active	-	-	Text	64
13	Length	Length of vessel (see NAIS specifications)	71.0	meters	XXX.X	Float	4
14	Width	Width of vessel (see NAIS specifications)	12.0	meters	XXX.X	Float	4
15	Draft	Draft depth of vessel (see NAIS specifications)	3.5	meters	XXX.X	Float	4
16	Cargo	Cargo type (see NAIS specification and codes)	70	-	-	Text	4
17	TransceiverClass	Class of AIS transceiver	AIS Class A	-	-	Text	16

Table 1. Products File Format

Example Line from Input File

```
367430850,2019-01-01T00:00:05,30.40366,-88.85801,0.0,126.3,511.0,MISS NIZ,IMO8987539,WDF2697,31,0,22,8,2.4,52,B
```

Prototypes for Student Supplied Functions

```
void readFile( ifstream & inFile, vector<AISType> &item, int& count);
bool openInputFile( ifstream & inFile );
string makeStringUpper( string s);
int searchForVesselByName( vector<AISType> & dataBase, string vesselName,
                           vector<string> & s );
void printRecord( AISType & item );
bool getNextField(string &line, int &index, string &subString);
double stringConvert(string);
int findLastOccurrance(string mmsi, vector<AISType> &d);
int findFirstOccurrance(string mmsi, vector<AISType> &d);
void addUniqueString( vector<string> &s, string value);
void saveField( int fieldNumber, string subString, AISType &tempItem );
double distanceTraveled( vector<AISType> & dataBase, int first, int last );
```

Detailed Function Descriptions

void readFile(ifstream & inFile, vector<AISType> &item, int& count);
inFile – already open file to be read
item – vector to store data
count – the number of records read
This function uses the previously opened filestream to read in the data from the file. You must use getline() to read the data from the file with a '\n' as the terminating character. Once the line of data has been read, use *getNextField()* to parse the line, one field at a time. The initial call after the *getline()* should have the index set to zero, so it starts at the beginning of the string. *getNextField()* will update the index, so it won't need to be reset to zero until after the next call to *getline()*. The data should be read until end of file is reached.   See example output.

The function should start with a locally declared and initialized *AISType* temporary record. As the line is read/parsed and the fields are parsed, set the corresponding value in the temporary record. Once parsing the line is complete, and the temporary record is filled, use *push_back( )* to add the temporary record to the *AISType* vector. Reset all values in the temporary record to default before reusing the temporary record on the next line.

The function must keep a counter of the number of records read. The counter is used for outputting the number or records read.

readFile() uses the following C++ functions
getline() - C++ library
getNextField() - student provided
saveField() - student provided

Algorithm for *readFile()*

1. use *getline()* to read the line into a string
2. check for *eof()*
3. set index to 0, *fieldNumber* to 0
4. call *getNextField()* on the string read in 1, passing index, and a string to receive the field
5. use *saveField()* to store the data in a temporary *struct* of type *AISType* (use *fieldNumber* to determine which field gets the data)
6. increment *fieldNumber*
7. loop back to 4 until the entire string has been processed.
8. add the temporary *struct* holding the data to the vector using *.push_back()*
9. reset *fieldNumber* and index to 0
10. loop back to 1 till eof()

To get the progress count, place the following code at the bottom of the read loop.

```
// Output the number processed. If the number update stops, then
// their may be problem with the program.
char eraseLine[]={'\r',27,'[','1','K'};

if( (count % 100000 ) == 0){
        cout << eraseLine << count ;
        cout.flush();
}
```

bool getNextField(string &line, int &index, string &subString)
string &line – the line of data read from the file that needs to be parsed
int &index – the current starting position of the parsing. The first time this function is called for a new line, index should be set to zero. The function should update the index before returning, so that on the next call it will look at the next field.
string &subString – the parsed string

This function will receive the line of data read from the file, and starting at *index*, find the next string, which is all characters upto the next comma or '\n'. Fields are delimited by commas. The string found, not including the comma, is saved in the *subString*. The commas that separate the data should be skipped. The value of *index* should be updated as the line is processed. When the function returns, *index* should point to the next character to be processed, and will be used as the starting point for the next call to the function. This function must parse the line one character at a time. String functions like *.find()* **may not** be used.

Pseudo Code for getNextField
(suggested, you may do it differently)

```
string line;
string str;

index = first character of next field
while not at end of string and line[index] != ‘,’)
  str += line[index];
  index++;
end while;
index ++;       // increment past the next ‘,’
                // getting ready for the next call getNextField

// test to see if there is more data in the string.
if line.length() <= index
    return false;
else return true
```

bool openInputFile(ifstream& inFile);
ifstream& infile – file stream variable for the file to be opened.

This function should prompt the user for the name of the csv file to open. If the user enters (q/Q) the function should return false. For any other input, the file should be opened. You must verify that the file opened correctly. If the file did not open, an informative message should be printed and the user should be re-prompted. If the file opens correctly, the function should return true.

void printRecord(AISType& item);
AISType& item - The record to print. This is a single *item*, not the vector.

The function prints all fields for the record. See example output for formatting.

int searchForVesselByName(vector<AISType> & dataBase, string vesselName,
vector<string> & matchesAIS);
vector<AISType> & dataBase – The database of AIS records read from the file
string vesselName – the string to be searched for in the vessel names.
vector<string> & matchesAIS – all vessel whose names contains the passed string, *vesselName,* should have their MMSI string added to the passed vector *matchesAIS*. *matchesAIS* is a vector that only contains the MMSI of vessels.

return value – the number of records found that match the passed string, vesselName.

This function performs a linear search on all records in *dataBase* for records whose *vesselName* member contains the string passed in *vesselName.* The search should be done from 0 to maximum valid index. This function calls *addUniqueString()* to add the MMSI to the *matchesAIS* vector.

string makeStringUpper(string s);
string s – the string to be converted to upper case.
return value – upper case version of passed string.

This function converts the passed string to upper case and returns it. The library function *toupper()* may be called by this function.

double stringConvert(string s)
string s – the string to be converted into a double

return value – the double converted from the string

This function converts a string to it's corresponding double value. Implementation of the function must use string stream.   All other implementations are not allowed and will have significant point deductions.

int findLastOccurrance(string mmsi, vector<AISType> &d)
string mmsi - the mmsi value being searched for
vector<AISType> &d – the vector being searched.

return value – the index of the last record in the vector that contains the *mmsi* value passed in.

This function performs a linear search to locate the last record in the passed vector with the specified MMSI. For efficiency, the search should be started at the last element and work toward the first.

int findFirstOccurrance(string mmsi, vector<AISType> &d)
string mmsi - the mmsi value being searched for
vector<AISType> &d – the vector being searched.

return value – the index of the first record in the vector that contains the *mmsi* value passed in.

This function performs a linear search to locate the first record, in the passed vector, with the specified MMSI. The search should be started at the first element and work toward the last.

void addUniqueString(vector<string> &s, string value)
vector<string> &s – the vector that the string should be added to
string value – the string to be added

This function first searches the vector to determine if the string in already present. If the string is already present, the function simply returns. If the string is not present, the it is added at the end of the vector. A string will never appear twice in the vector.

void saveField(int fieldNumber, string subString, AISType &tempItem)
int fieldNumber – the number of the field, starting at zero
string subString – the value to be saved in the field, may require conversion to double inside the function.
AISType &tempItem- the record to which the field will be added

This function saves the *subString* in to *fieldNumber* in the record passed, *tempItem.* The *subString* may need to be converted to a *double,* depending on the *fieldNumber.* See the definition of *struct AISType* for specific field data types. This function must use *stringConvert()* to perform the conversion from *string* to *double*. A error message should be output from the function if an unknown field number is specified.

double distanceTraveled(vector<AISType> & dataBase, int first, int last)
vector<AISType> & dataBase - vector containing the AIS data records
int first – index of starting location record
int last – index of ending location record

return value – calculated distance. In the case of a bad index value, 0.0 should be returned.

Use the haversine formula to calculate the great circle distance between the vessel location in records *first* and *last*, using latitude and longitude. Remember that C++ functions, sine, cosine, and arc tangent require the input values to be in radians.

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{(1-a)})$$
$$distance = R \cdot c$$
$$\varphi\ is\ latitude,\ \lambda\ is\ longitude$$
$$R = 3958.8(mean\ radius\ of\ earth, miles)$$
$$PI = 3.14159$$
$$radians = (degrees * PI)/180.0$$

Formulas are from the following website.
Calculate distance website

This website may also be used to verify that your distances are correct. All website distances are in km, while program distances are in miles.

Example Output

```
dolly@Snoopy:Assignment 1$ g++ main_solution.cpp -Wall -pedantic
dolly@Snoopy:Assignment 1$ ./a.out
Enter input File Name/(q-quit): AISData.csv
File opened correctly
----------------------
1800000--- End of file reached ---Items read: 1899355
1899355 records read
Enter vessel name: bill

Searching for records with names containing "bill"

1543 vessels found with name containing "bill", Unique vessels: 2

2 vessels found. Would you like to see their first records? [y/n] y
************************
MMSI:          367007060
Base Date Time: 2019-01-01T00:00:00
Lattitude: 30.2898 Longitude: -91.2234
SOG: 0.2
COG: -196.3
Heading: 511
Vessel Name: BILL S
imo: IMO7030963
Call Sign: WDC3383
Vessal Type: 31
Status: 15
Length: 30
Width: 8
Draft: 3.6
Cargo: 0
Transceiver Class: B


************************
MMSI:          366751770
Base Date Time: 2019-01-01T11:27:05
Lattitude: 49.9244 Longitude: -125.12
SOG: 6.5
COG: 131.1
Heading: 132
Vessel Name: BILLIE H
imo: IMO8964719
Call Sign: WCY4992
Vessal Type: 31
Status: 0
Length: 27
Width: 8
```

```
Draft: 4.7
Cargo: 32
Transceiver Class: B


Would you like to find the distance traveled for a vessel? [y/n] y
MMSI for vessel: 366751770

Vessel: "BILLIE H" MMSI: 366751770  Trip Starting time: 2019-01-01T11:27:05
Distance traveled from (49.9244, -125.12) to (49.1369, -123.534) 89.5485 Miles

Enter vessel name: maersk

Searching for records with names containing "maersk"

15706 vessels found with name containing "maersk", Unique vessels: 29

29 vessels found. Would you like to see their first records? [y/n] n
Enter vessel name: atomic

Searching for records with names containing "atomic"

363 vessels found with name containing "atomic", Unique vessels: 1

1 vessels found. Would you like to see their first records? [y/n] y
************************
MMSI:         319071600
Base Date Time: 2019-01-01T00:04:11
Lattitude: 26.7489 Longitude: -80.0499
SOG: 0
COG: 186.5
Heading: 182
Vessel Name: ATOMIC
imo: IMO1009807
Call Sign: ZGEG8
Vessal Type: 37
Status: 5
Length: 45
Width: 9
Draft: 2.8
Cargo:
Transceiver Class: A


Would you like to find the distance traveled for a vessel? [y/n] y
MMSI for vessel: 319071600

Vessel: "ATOMIC" MMSI: 319071600  Trip Starting time: 2019-01-01T00:04:11
Distance traveled from (26.7489, -80.0499) to (26.7489, -80.0499) 0.00151337 Miles

Enter vessel name: mldf

Searching for records with names containing "mldf"
Vessel "mldf" not found
Enter vessel name: q
```